# RNN (LSTM vs GRU):

* A recurrent neural network, by contrast, retains a memory of what it has processed in its recent previous steps (we’ll come back to the “recent” qualifier in a minute). It makes recurrent connections by going through temporal feedback loops: the output of a preceding step is used as an input for the current process step. Unlike amnesiac FFNNs, this memory enables RNNs to process sequences of inputs without losing track. The loops make it a recurrent network.
* to handle sequential data successfully, you need to use recurrent (feedback) neural networks. It can ‘memorize’ parts of the inputs and use them to make accurate predictions. These networks are at the heart of speech recognition, translation, and more.
* **“Vanishing gradient problem” in standard RNN**: the problem comes from the fact that at each time step during training, we are using the same weights to calculate y\_t. That multiplication is also done during back-propagation. The further we move backward, the bigger or smaller our error signal becomes. This means that the network has trouble in memorizing words from far away in the sequence and makes predictions based on only the most recent ones.
* The key difference between a GRU and an LSTM is that a GRU has two gates (reset and update gates) whereas an LSTM has three gates (namely input, output, and forget gates). Why do we make use of GRU when we clearly have more control on the network through the LSTM model (as we have three gates)? In which scenario GRU is preferred over LSTM?
  + GRUs and LSTMs utilize different approaches toward gating information to prevent the vanishing gradient problem. Here are the main points comparing the two:
    - The GRU unit controls the flow of information like the LSTM unit, but **without having to use a memory unit**. It just exposes the full hidden content without any control.
    - GRUs are relatively new, and in my experience, their performance is on par with LSTMs, but computationally more efficient (as pointed out, they have a less complex structure). For that reason, we are seeing it being used more and more.
    - GRUs train faster and perform better than LSTMs on less training data if you are doing language modeling (not sure about other tasks).
    - GRUs are simpler and thus easier to modify, for example adding new gates in case of additional input to the network. It's just less code in general.
    - LSTMs should in theory remember longer sequences than GRUs and outperform them in tasks requiring modeling long-distance relations.

# RNN working: [Credits](https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21)

In RNN, tanh is used as the activation function.

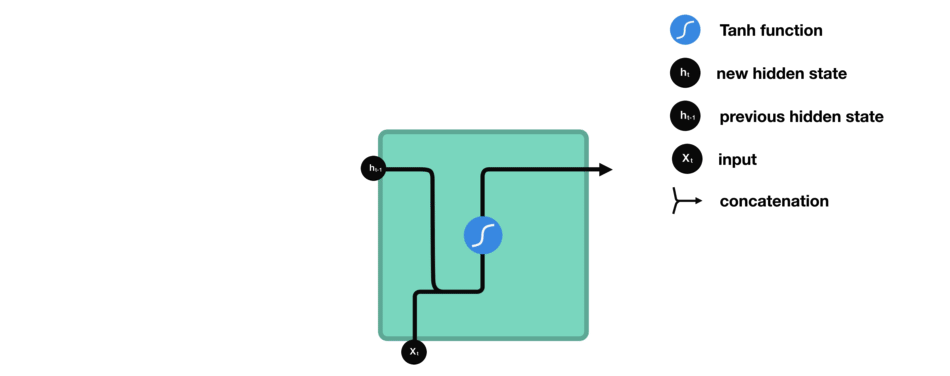


Figure 1: Tanh Credits: <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>

Tanh activation function scales the range of the values to range between -1 and 1. This is performed so that the mathematical operations in the neural network don’t blow up these values into huge numbers.

## LSTM (Long Short Term Memory)

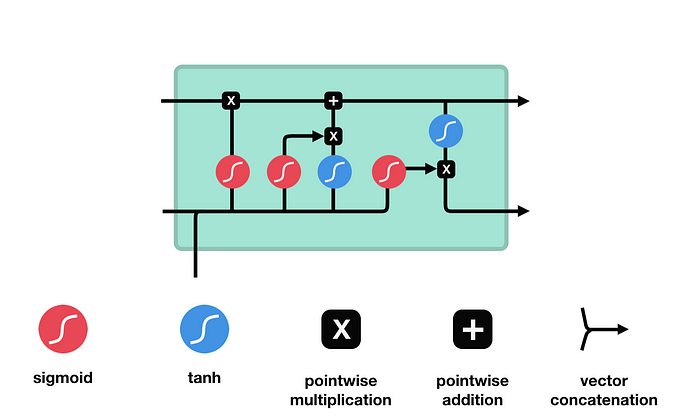


Figure Credits: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

In LSTM, the Sigmoid function is also used as an activation as it similar to tanh, scales the values between 0 and 1. That is helpful to update or forget data because any number getting multiplied by 0 is 0, causing values to disappear or be “forgotten.” Any number multiplied by 1 is the same value therefore that value stay’s the same or is “kept.” The network can learn which data is not important therefore can be forgotten or which data is important to keep.

### Forget Gate

Sigmoid Function which gives out a value which either 0 or 1. 0 means forget, and 1 means keep.

### Input gate

We have the input gate to update the cell state. The hidden state and input are passed through the tanh function to scale them. These 2 are parallelly also passed through the sigmoid function to get 0 or 1 (not important or important). These 2 outputs are then multiplied and so the sigmoid will decide which information from the tanh is important.

### Cell State

The cell state is first pointwise multiplied with the forget gate output and so the irrelevant points are dropped. Then the output from the input gate is pointwise added to this state to update the values to new values that the NN finds relevant. This is our new cell state.

### Output Gate

Hidden states carry information from previous states and are usually used for prediction. First, the information from the previous hidden state and current input are passed through the Sigmoid function. Next, the new cell state is passed through the tanh function and is then multiplied with the output of the sigmoid to determine what information is important which gives the new hidden state.

### Summary

The Forget gate decides what is relevant to keep from prior steps. The input gate decides what information is relevant to add from the current step. The output gate determines what the next hidden state should be.

## GRU (Gated Recurrent Unit)

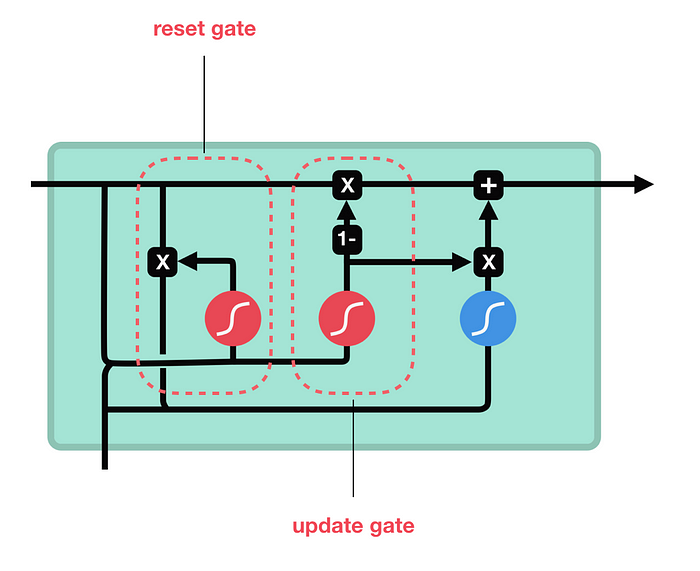


Figure Credits: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

### Update Gate

The update gate acts similarly to the forget and input gate of an LSTM. It decides what information to throw away and what new information to add.

### Reset Gate

The reset gate is another gate used to decide how much past information to forget.

### Summary

GRU has 2 states and is therefore faster than LSTM.

## Summary

A model can be chosen based on the use case.